VŠB TECHNICKÁ UNIVERZITA OSTRAVA
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Absolvování individuální odborné praxe

Individual Professional Practice in the Company

Marek Chutný

Bakalářská práce

Vedoucí práce: Ing. Petr Lukáš, Ph.D.

Ostrava, 2021

Abstrakt

Tato práce popisuje průběh mého působení ve firmě ComProMiS s.r.o., ve které jsem absolvoval odbornou praxi. Hlavním předmětem této praxe je elektronicky docházkový systém. Začátek práce je věnován představení firmy, následuje popis zadaných úkolu, fungování aktuálního docházkového systému a představa nového. V následujících kapitolách je popsán návrh webové aplikace, funkčnost a použité technologie k vývoji. Další text pojednává o samotném vývoji webové aplikace. V poslední části je sepsáno zveřejnění, testování a ladění této aplikace. Práce končí shrnutím znalostí, jenž jsem nabyl v průběhu studia, které byly využity během praxe, ale také těch, které mi scházely a celkovým zhodnocením praxe.

Klíčová slova

ASP.NET, MVC, C#, ComProMiS, odborná praxe, webová aplikace, docházkový systém

Abstract

This bachelor thesis describes the course of my work in the company ComProMiS s.r.o., in which I completed professional practice. The main goal of this practice is an electronic attendance system. The beginning of the work is devoted to the introduction of the company, followed by a description of the assigned tasks, the functioning of the current attendance system and the idea of a new one. The following chapters describe the design of the web application, the functionality and the technologies used for development. The next text deals with the development of the web application itself. The last part describes the publication, testing and debugging of this application. The work ends with a reduction of knowledge, I only gained during the study, which were used during the internship, but also those that I lacked and the overall evaluation of the internship.

Keywords

ASP.NET, MVC, C#, ComProMiS, individual practise, web application, attendance system

Poděkování

Rád bych poděkoval Ing. Romanu Potocznému za příležitost absolvování odborné praxe ve firmě ComProMiS s.r.o.

Poděkování patří také zaměstnancům této firmy, zejména Bc. Michalu Hrdému, Ing. Danielu Hrtúsovi a Bc. Janu Vožickému za odborné vedení, rady, při řešení problému a čas, který mí věnovali po celou dobu vykonávání odborné praxe.

V poslední řadě bych chtěl poděkovat Ing. Petru Lukášovi, Ph.D. za ochotu a jeho věcné připomínky při psaní této bakalářské práce.

Obsah

Seznam použitých zkratek a symbolů

CSS	–	Cascading Style Sheets
HTML	–	Hyper Text Markup Language
LINQ	–	Language Integrated Query
API	–	Application Programming Interface
ORM	–	Objektově relační mapování

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

Hlavním účelem této práce je přiblížit mnou absolvovanou praxi ve firmě ComProMiS s.r.o. v rámci posledního ročníku bakalářského studia. Úkolem bylo vytvořit nový elektronický docházkový systém, který je v dnešní době důležitý pro všechny zaměstnavatele. Moderní elektronické docházkové systémy mají nespočet výhod oproti starým systémům, kdy se zaměstnanec musí zapisovat do knihy nebo používat čipové karty. U těchto typů docházkových systémů může dojít ke zneužívání, kdy se zaměstnanci mohou navzájem zapisovat do knihy příchodů a odchodů, případně si mezi sebou půjčovat jíž zmiňované čipové karty. Moderní elektronické docházkové systémy umožňují zaměstnavatelům nejenom snadné sledování aktivit zaměstnanců, ale také jiné důležité náležitosti spjaté s docházkou. Práce na novém docházkovém systému probíhala v týmu, ve kterém jsem měl na starost vývoj webové aplikace.

Kapitola 2 představuje firmu ComProMiS s.r.o., popisuje její zaměření ve světě informačních technologií a přibližuje mé pracovní zařazení, pozici a tým, kterého jsem byl součástí. Následující kapitola 3 pojednává o zadaných úkolech práce na projektu spolu s jejich časovou náročností při plnění praxe ve firmě. Další kapitola 4 ukazuje stávající docházkový systém této firmy, pracovní povinnosti zaměstnance, problematiku dovolené, měsíční kontrolu docházky a další specifické sitauce. Kapitola 5 obsahuje představu o novém docházkovém systému. Důvodem jeho vzniku je zjednodušení funkce stávajícího docházkového systému, nejen z pohledu zaměstnance, ale také z pohledu účetního, který má za úkol počítat měsíční mzdy. Kapitola 6 představuje přibližný návrh zobrazení jednotlivých obrazovek webové aplikace. Dále zde definuje uživateské role a jejich přístupová práva. Následuje kapitola 7, která ukazuje popis všech zvolených technologií k vývoji webové aplikace. Ke každé technologii je uveden teoretický úvod. Hlavní část této práce v kapitole 8 popisuje samotný vývoj webové aplikace, která byla také časově nejvíce náročná. K této části se váže i další kapitola 9, která pojednává o nasazení a testování aplikace. Práci ukončuje shrnutí znalostí, které jsem nabyl v průběhu studia a které byly využity během praxe ale také těch, které mi scházely. Posledním bodem je celkové zhodnocení této praxe.

Kapitola 2

Představení firmy a pracovní zařazení

2.1 ComProMiS s.r.o.

Společnost ComProMiS s.r.o. je na trhu již dvacet pět let a sídlí v Ostravě-Kunčičkách. Specializuje se na zakázkovou tvorbu software a údržbu specializovaných informačních systémů. Jednou z předností firmy je dlouhodobá zkušeností s tvorbou těchto systémů, webových stránek a aplikací pro mobilní zařízení. V současné době se soustřeďuje převážně na vývoj v prostředí operačního systému Windows a v moderních techonologiích na bázi .NET Framework, jako jsou ASP.NET, MVC, C# a Java v databázových prostředích Microsoft SQL Server nebo Oracle. Většina vyvíjených aplikací směřuje do prostředí podpory nákupu, zpracování zakázek a výměny informací mezi centrálami a pobočkami. Nemalou část aktivit firmy tvoří i tvorba aplikací pro podporu servisních aktivit, zejména pro světovou společnost OTIS zabývající se výtahy, s níž udržují dlouhodobou spolupráci. [1]

2.2 Pracovní zařazení

Společnost ComProMiS s.r.o. jsem si vybral z pestré nabídky firem, které nabízí škola, kde mohou studenti posledních ročníků bakalářského studia vykonat odbornou praxi. Poté proběhl v sídle společnosti krátký pohovor. Do firmy jsem byl přijat na pracovní pozici analytik-programátor webové aplikace. Připojil jsem se k vývojovému týmu, který měl za úkol vytvořit nový docházkový systém firmy. Tým se skládal ze dvou vývojářů aplikací pro mobilní systémy Android a iOS, vývojáře API a mě, jako vývojáře webové aplikace. Celý tým byl složen ze studentů VŠB. Po celou dobu trvání praxe probíhaly pravidelné konzultace se zaměstnanci firmy, kteří prováděli kontrolu odvedené práce a předávali mi cenné rady. Vzhledem k mimořádné situaci spojené s pandemií Covid-19 v roce 2020/2021 byla firma nucena přejít na formu dálkového pracoviště, tudíž velká část práce vznikla v místě bydliště a konzultace probíhaly prostřednictvím telekonferencí.

Kapitola 3

Zadání práce projektu

Již v minulé kapitole jsem nastínil, že předmětem této práce je vývoj docházkového systému. V následujících bodech rozebírám po částech, jaké byly úkoly při jeho vývoji. Časovou náročnost těchto úkolů vyobrazuje tabulka 3.1.

1. **Porozumění stávajícího systému ve firmě** – V počáteční fázi praxe jsem od vedoucího obdržel dokument s přehledem toho, jak aktuální docházkový systém přibližně vypadá. Jeho stručný přepis, který jsem si vytvořil a následně používal i k programování aplikace, můžeme najít v kapitole 4.

2. **Představa nového systému, návrh a tvorba databázového modelu** – Další týdny praxe jsme se s mým týmem věnovali, tomu, jak budeme celý systém chtít koncipovat, a navrhovali různé verze databázového modelu. Podrobnosti k tomuto bodu rozepisuji v kapitole číslo 5, kde se nachází i finální návrh databázového modelu spolu s jeho popisem.

3. **Seznámení s Microsoft Azure a tvorba struktury projektu v Azure DevOps** – Rozhraní Azure mi bylo doporučeno ze strany firmy, protože v ní firma sama zpracovává velké množství projektů.

4. **Návrh zobrazení a funkčnost aplikace** – Před začátkem samotného vývoje aplikace, bylo důležité navrhnout strukturu hlavní nabídky spolu s návrhem některých obrazovek aplikace a jejich rozložení. Detailnější popis spolu s obrázky návrhu můžeme najít v kapitole 6 .

5. **Vývoj webové aplikace technologií ASP .NET MVC** – Webová aplikace byl hlavní a taky časově nejnáročnější úkol celé mé praxe. Tento bod tvorby je popsán detailněji v kapitole 8.

6. **Zveřejnění, testování a ladění** – Posledním úkolem bylo zajistit snadný přístup k webové aplikaci. Tuto část rozebírám v kapitole 9.

Úkol	**Časová náročnost**
Porozumění stávajícího systému ve firmě	2 dny
Představa nového systému, návrh a tvorba databázového modelu	6 dny
Seznámení s Microsoft Azure a tvorba struktury projektu v Azure DevOps	2 dny
Návrh zobrazení a funkcí aplikace	2 dny
Vývoj webové aplikace technologií ASP .NET MVC	28 dní
Zveřejnění, testování a ladění	10 dny

Tabulka 3.1: Časová náročnost jednotlivých úkolů

Kapitola 4

Stávající systém

Tato kapitola pojednává o tom, jakým způsobem zaměstnanci evidují svou pracovní přítomnost nebo nepřítomnost a popisuje vnitřní pracovní povinnosti firmy.

4.1 Pracovní doba

Povinná pracovní doba firmy je od 09:00 do 14:00 hodin, kdy v tuto dobu zaměstnanec musí pracovat. Každý zaměstnanec musí plnit tuto povinnou pracovní dobu, pokud tak neučiní, je povinen čerpat půldenní dovolenou. Příchody i odchody se zaokrouhlují po 15-ti minutách. Příchody se zaokrouhlují nahoru (pokud zaměstnanec přichází v 9:05, práce je evidována až od 9:15) a odchody dolů (pokud zaměstnanec odchází 14:19, jeho pracovní doba končí v 14:15). Jestliže odpracovaná doba překročí 5 hodin v den, kdy zaměstnanec pracuje v místě firmy, vzniká zaměstnanci nárok na půlhodinovou přestávku, která se nezapočítává do odpracované doby. Rovněž automaticky vzniká nárok na příspěvek stravování ve formě stravenky.

4.2 Pracovní úvazky

Zaměstnanci mají smluvně daný pracovní úvazek, který může být denní nebo týdenní.

Denní úvazek - Zaměstnanec odpracuje každý pracovní den určitý počet hodin. Ve firmě ComProMiS s.r.o. je tento úvazek nastaven na 7,5 hodin denně.

Týdenní úvazek - Zaměstnanec odpracuje každý týden určitý počet hodin. Ve firmě ComProMiS s.r.o. je tento úvazek nastaven na 37,5 hodin týdně. Zaměstanci s tímto typem úvazku musí odpracovat povinnou pracovní dobu a zbytek hodin si mohou rozvrhnou libovolně mezi další pracovní dny v týdnu.

4.3 Pracovní neschopnost zaměstnance

Zaměstnavatel musí omluvit nepřítomnost zaměstnance v práci podle zvláštních právních předpisů po celou dobu jeho dočasné pracovní neschopnosti. U těchto překážek v práci je zaměstnanec povinen prokázat se platným dokladem o pravdivosti. Mezi tyto překážky patří:

- **Lékař** - Pokud zaměstnanec navštíví lékaře, započítává se tento čas do odpracované doby. Pokud se jedná o celodenní návštevu, počítá se tento den jako odpracovaný.
- **Nemocenská** - Čas, který zaměstnanec stráví na nemocenské se odečítá od hodin pracovního úvazku.
- **Ostatní důvody** - Tyto důvody můžeme najít v zákoníku práce [2] a řeší se obdobně jako lékař.

4.4 Specifické situace

- **Státní svátek** - Jestliže státní svátek připadne na pracovní den, počítá se tento den jako odpracovaný.
- **Dovolená** - Zaměstnanci firmy mají nárok na 25 dnů dovolené ročně. Pokud zaměstnanec nevyčerpá dovolenou pro daný rok, určitý počet si může přenést do roku nového. Stejně jako u nemocenské se čas strávený na dovolené odečítá od hodin pracovního úvazku.
- **Home Office** - Zaměstnanci mohou dle směrnic firmy pracovat z místa bydliště, případně jiného místa nahlášeného zaměstnavateli. Home office se počítá jako odpracovaný den.
- **Pracovní cesta** - Počítá se jako odpracovaný den.

Ve stávajícím docházkovém systému si zaměstnanci plánují dovolené, home office a pracovní cesty do firemního kalendáře na webu.

4.5 Stávající zápis a výpočet docházky

V současné době zaměstnanci evidují své příchody a odchody do knihy. Všichni zaměstnanci jsou sami zodpovědní za zápisy do této knihy. Kontrola, zda zaměstnanci odpracovali kompletní počet hodin, probíhá na konci každého měsíce. Asistentka si z knihy spočítá součet odpracovaných hodin z příchodů a odchodů pro každého zaměstnance. Tyto hodnoty si zapíše do tabulky v programu Microsoft Excel, přičemž musí v kalendáři firmy zkontrolovat všechny specifické situace pro zaměstnance v daný měsíc, které si pak porovnává s odpracovanými hodinami.

Kapitola 5

Nový systém

V této kapitole popisuji představu nového docházkového systému firmy. Dále zde rozvedu návrh databázového modelu. Prototyp tohoto systému již existoval, ale kvůli nedostatečné funkcionalitě a změnám, které za tu dobu nastaly, byl tvořen zcela od začátku.

5.1 Představa

Nový elektronický docházkový systém by měl nahradit ten stávající a bude sloužit pro evidenci všech aktivit a plánů zaměstnance spolu s možností jejich správy. Plány budou v systému prezentovány jako směrnice pracovních a nepracovních aktivit (například nemocenská, dovolená, státní svátek, home office) pro daný den. Tento systém se má skládat z mobilních aplikací (pro iOS a Android), rozhraní web API a webové aplikace. Pro přístup do systému budou mít všichni zaměstnanci firmy vytvořen účet.

5.1.1 Mobilní aplikace

Mobilní aplikace by měly pracovat na operačním systému Android a iOS. Všichni zaměstnanci budou mít aplikaci nainstalovanou ve svém mobilním zařízení. Pokud zaměstnanec nebude mít chytrý telefon, je na domluvě se zaměstnavatelem, jakým způsobem bude aktivity vykazovat.

Tyto aplikace budou převážně sloužit k evidování aktivit - začátku a konce práce s možností přerušení a vytvoření nové aktivity (například lékaře nebo obědové pauzy). Začít a ukončit práci aplikace povolí pouze v případě, kdy je zaměstnanec přímo na místě, nebo blízko místa pracoviště, proto bude mít aplikace přístup k poloze zařízení, jestliže uživatel aplikaci přístup k poloze nepovolí, aplikace se nespustí. Proběhlé aktivity si zaměstnanec bude moct prohlédnout v detailu. Aplikace budou řešit také žádosti, ve kterých budou vznikat požadavky na plánované dovolené, nemocenské, pracovní cesty a podobně. Aplikace také bude vytvářet místa pro vykonávání aktivit z home office.

5.1.2 Webová aplikace

Webová aplikace by měla sloužit jako administratorská část celého systému. K dispozici bude na serveru kvůli snadné možnosti přístupu z webového prohlížeče. Nejdůležitější funkcí této aplikace má být schvalování docházky. To znamená, že aplikace bude muset kontrolovat, zda byl naplněn úvazek zaměstnance a kontrolovat všechny pracovní povinnosti. Další funkcí má být zobrazování měsíčních plánů, jejich vytváření nebo mazání. S tím souvísí funkce, která bude vytvářet přehledy na měsíční bázi, kde budou právě tyto plány znázorněny spolu se skutečně odpracovanými hodinami a hodinami, které zaměstnanec měl odpracovat dle plánu. Nedílnou součástí bude přehledné zobrazování požadavků zaměstnanců, jejich schvalení či zamítnutí s možností zobrazení detailu.

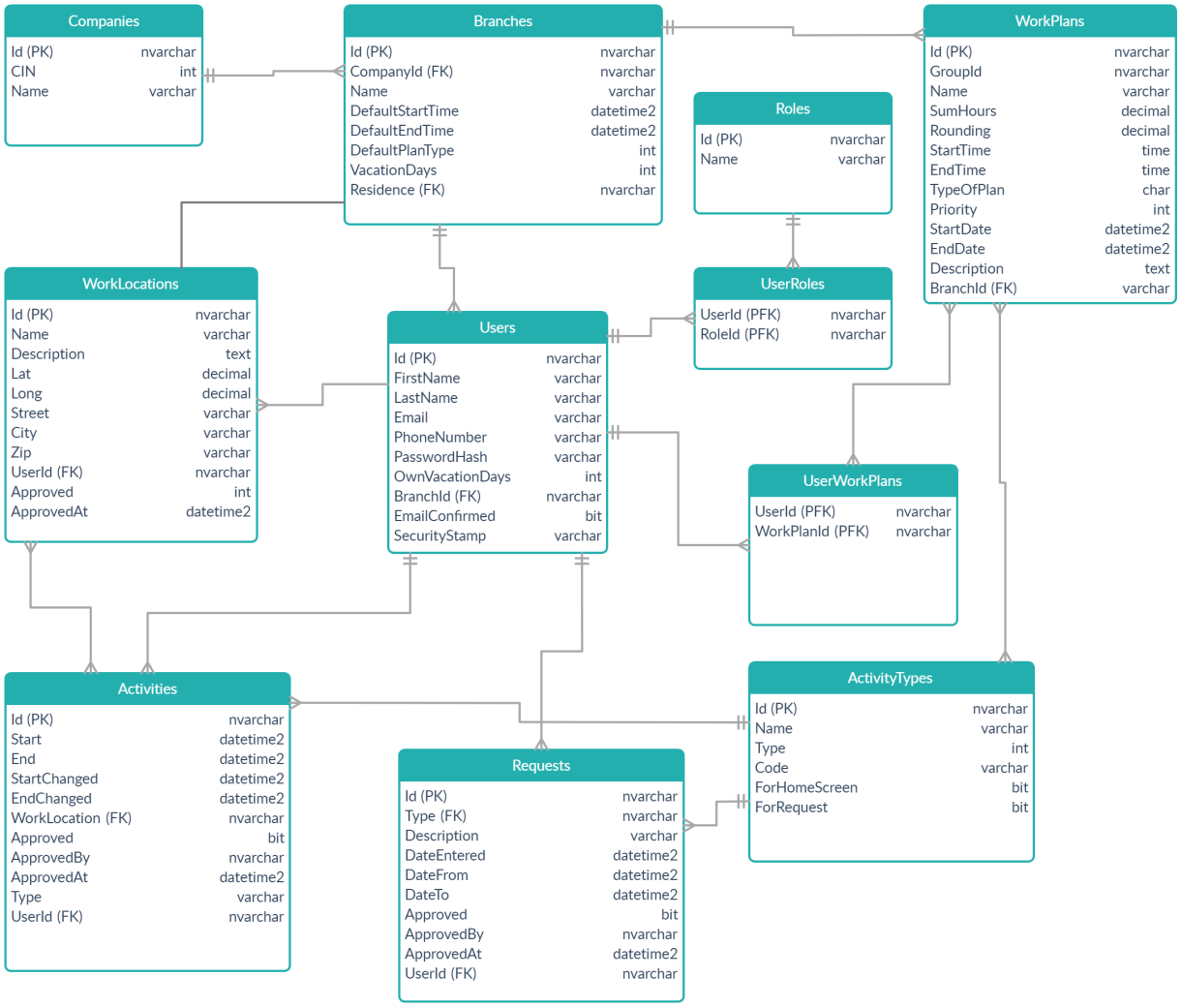
5.1.3 Rozhraní Web API

Web API rozhraní by mělo mít za úkol provádět komunikaci mezi databází a mobilními aplikacemi. Budou zde prováděny všechny důležité výpočty a operace, které poté budou používat mobilní aplikace.

5.1.4 Databáze

Databáze bude sloužit k uchovávání informací, které budeme potřebovat v rámci celého docházkového systému. Po konzultaci s kolegy z firmy, jsme se rozhodli využít databázový systém Microsoft SQL Server.

5.2 Databázový model



Obrázek 5.1: Databázový model projektu

5.2.0.0.1 Companies Obsahuje pouze základní informace o firmě jako korporátní identifikační číslo nebo název.

5.2.0.0.2 Branches Tato tabulka má v sobě atributy, které určují základní detaily pobočky. Za zmínku zde stojí atributy `DefaultStartTime`, `DefaultEndTime` a `DefaultPlanType`. Tyto atrbibuty definují začátek a konec pracovní doby pobočky a taky základní typ úvazku.

5.2.0.0.3 WorkPlans Jedna z nejdůležitějších tabulek databázového modelu. Obsahuje definici pracovních plánů, které budou sloužit jako směrnice aktivit pro daný den. Tyto plány jsou přiděleny zaměstnancům a pobočkám. Podstatným atributem je zde atribut `Priority`, který nabývá hodnot 0, 1, 2 nebo 3.

- **Hodnota 0** - Označuje všechny základní typy úvazků.
- **Hodnota 1** - Označuje státní svátky.
- **Hodnota 2** - Označuje všechny plány, které jsou celopobočkové nebo celofiremní.
- **Hodnota 3** - Označuje speciální plány zaměstnance (například dovolené).

5.2.0.0.4 UserWorkPlans Slouží jako vazební tabulka mezi tabulkami `WorkPlans` a `Users`. Jeden pracovní plán může mít více zaměstnanců a jeden zaměstnanec může mít přiřazeno více plánů.

5.2.0.0.5 Users Určuje základní informace o uživateli dané pobočky jako je jeho jméno (`Name`), email, telefon (`PhoneNumber`) případně počet zbývajících dnů dovolené (`OwnVacationDays`). Dále se v této tabulce náchází atributy související se zabezpečením uživatelského účtu.

5.2.0.0.6 Roles Tabulka nesoucí informace ohledně rolí v systému.

5.2.0.0.7 UserRoles Slouží jako vazební tabulka mezi tabulkami `Roles` a `Users`. Jedna role může mít přiřazeno více uživatelů a jeden uživatel může mít více rolí.

5.2.0.0.8 WorkLocations Tabulka, díky které bude fungovat ověřování lokace uživatele v mobilní aplikaci. V této tabulce najdeme všechny lokace a místa. Hlavními atributy pro funkčnost ověření lokace jsou atributy `Lat` a `Long` udávající GPS pozice, tj. zeměpisnou šířku a délku.

5.2.0.0.9 Activities Ústřední tabulka celého systému. Záznamy v této tabulce popisují aktivity zaměstnanců. Obsahuje údaje o začátku (`Start`), konci (`End`), jejich změn (`StartChanged`, `EndChanged`), místo výkonu (`WorkLocation`) a také typu (`Type`). Aktivita se dle atributu `Approved` může nacházet v následujících stavech:

- **Approved** – Schválení.
- **ApprovedWithChanges** - Schválení se změnami.
- **Rejected** – Zamítnutí.
- **Pending** – Čekající.

5.2.0.0.10 Requests Popisuje požadavky zaměstnance a informací těchto požadavků jako je typ požadavku (`Type`), popis (`Description`), datum přídání (`DateEntered`) a status schávalení (`Approved`). Na základě schálení těchto požadavků budou vznikat i pracovní plány.

5.2.0.0.11 ActivityTypes Obsahuje záznamy definující typy aktivit. Důležité u této tabulky je povšimnou si atributu `ForHomeScreen` a `ForRequest`, tyto atributy nám říkají, zda nám typy aktivit budou k dispozici, při vytváření nových aktivit v mobilním zařízení nebo při vytváření požadavků zaměstnancem. Typ aktivity se dle atributu `Type` může nacházet v následujících stavech:

- **Productive** - Určuje všechny pracovní činnosti (například práce ve firmě, home office).
- **NotProductive** - Určuje všechny činnosti zaměstnance, které jsou omluveny zaměstnavatelem a počítají se jako odpracovaný čas (například návštěva lékaře).
- **Absence** - Učuje všechny činnosti, kdy zaměstnanec není povinen pracovat (například dovolená nebo nemocenská).

Kapitola 6

Návrh webové aplikace

Tato kapitola pojednává o návrhu rozložení webových stránek aplikace a jejich funkcionalitě. Implementace těchto stránek bude podrobně rozebrána v kapitole 8.

6.1 Funkčnost aplikace

Funkčnost aplikace by měla být rozdělena do několika rolí, které byly diskutovány s vedoucím praxe.

- **System-Admin** – Spravuje typy aktivit, generuje globálně státní svátky pro všechny uživatele tohoto systému.
- **Company-Admin** – Může vytvářet, upravovat nebo mazat pobočky firmy s možností přidělování a editace jejich zaměstnanců. Dále může plánovat například celozávodní dovolené firmy.
- **Branch-Admin** – Podle přidělené pobočky, může schvalovat aktivity a žádosti zaměstnanců. Dále také spravuje plány pobočky nebo zaměstnance. V poslední řadě vytváří měsíční přehledy.

6.2 Hlavní nabídka, popis stránek

Webová aplikace se bude skládat z několika stránek, které budou popsány v následujících podkapitolách. Vzhled a rozložení těchto stránek bylo konzultováno s vedoucím praxe. Samotná implementaci některých z těchto stránek, případně jejich částí bude dále popsána v kapitole 8. Každá podkapitola značí položku hlavní nabídky, která bude umístěná v horní částí všech stránek.

6.2.1 Přehled (Dashboard)

Přístup pro role: Company-Admin, Branch-Admin
Bude sloužit jako domovská obrazovka pro přihlášené uživatele, kde naleznou základní informace firmy nebo pobočky.

6.2.2 Docházka (Attendance)

Přístup pro role: Branch-Admin
Přehled aktivit za týden nebo den zaměstnanců pobočky. Zde bude umožněno tyto aktivity upravovat, mazat, přidávat a poté i schvalovat. Návrh obrazovky můžeme vidět na obrázku 6.1c.

6.2.3 Plánování (Planning)

Přístup pro role: Branch-Admin
Přehled plánů všech zaměstnanců pobočky s možností spravování. Bude zde možnost, skrýt víkendy, případně ukázat pouze plány jednoho zaměstnance. Na této obrazovce budou i požadavky na nové plány zaměstnanců (například dovolené) čekající na schválení s možností jejich schválení.

6.2.4 Požadavky (Requests)

Přístup pro role: Branch-Admin
Přehled všech požadavků od zaměstnanců pobočky, kde bude možné je schválit či zamítnout, případně rozklidnout detail, kde budou bližší informace. Tyto požadavky bude možné filtrovat podle zaměstnance, stavu schválení nebo jejich typů. Návrh této obrazovky můžeme vidět na obrázku 6.1d.

6.2.5 Firma (Company)

Přístup pro role: Company-Admin
Část hlavní nabídky obsahující rozbalovací seznam, ve kterém jsou následující stránky:

- Pobočky (Branches) - Administrace všech poboček firmy. Filtrování podle názvu pobočky a její adresy.
- Zaměstnanci (Employees) - Administrace zaměstnanců všech poboček. Možnost vyhledat uživatele podle jména nebo emailu.
- Plánování (Planning) - Přehled plánů všech poboček s možností plánovat firemní akce (například celozávodní dovolená).

6.2.6 Branch (Pobočka)

Přístup pro role: Branch-Admin
Část hlavní nabídky obsahující rozbalovací seznam, ve kterém jsou následující stránky:

- Zaměstnanci (Employees) - Administrace všech zaměstnanců pobočky, kterou uživatel spravuje (je dáno přiřazenou pobočkou a rolí Branch-Admin). Možnost vyhledat uživatele podle jména nebo emailu. Návrh obrazovky můžeme vidět na obrázku 6.1a.

- Plánování (Planning) - Přehled plánů pobočky a firmy s možností plánovat pouze pobočkové akce (například rekonstrukce). Návrh obrazovky můžeme vidět na obrázku 6.1b.
- Typy úvazků (Types of employment) - Administrace typů pracovních úvazku.

6.2.7 Sytém (System)

Přístup pro role: System-Admin
Část hlavní nabídky obsahující rozbalovací seznam, ve kterém jsou následující stránky:

- Typy aktivit (Activity types) - Administrace typů aktivit systému.
- Státní svátky (Public holidays) - Administrace státních svátků v systému.

(a) Návrh obrazovky „Employees“

(b) Návrh obrazovky „Branch/Planning“

(c) Návrh obrazovky „Attendance“

(d) Návrh obrazovky „Requests“

Obrázek 6.1: Návrhy obrazovek

Kapitola 7

Použité technologie

Tato kapitola se zabývá nejdůležitějšími technologiemi a nástroji, které jsem použil k vývoji webové aplikace.

7.1 Microsoft Azure

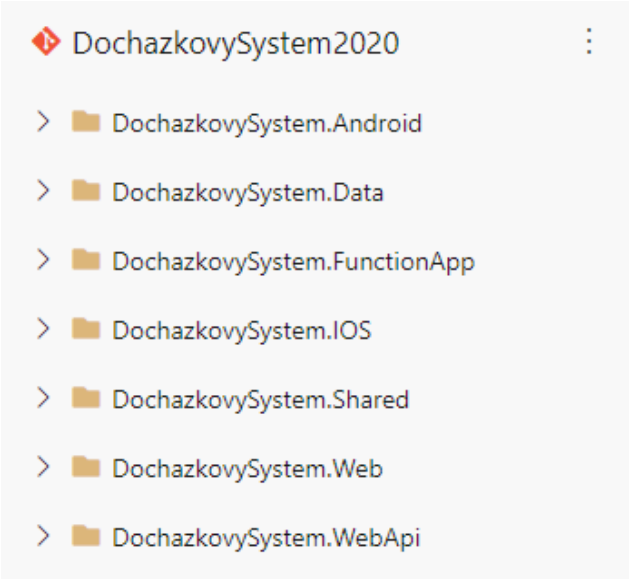
Azure je cloudové rozhraní, které je vyvíjeno společností Microsoft. Jak bylo již řečeno, toto rozhraní je založeno na principu cloudu. To znamená, že všechny produkty jsou uloženy na serverech a zákazníci k nim mohou přistupovat dálkově a odkudkoli, kde je Internet. Cloud nabízí nejrůznější řadu služeb zaměřujících se na vývoj, správu a hostování aplikací.

7.1.1 Azure DevOps

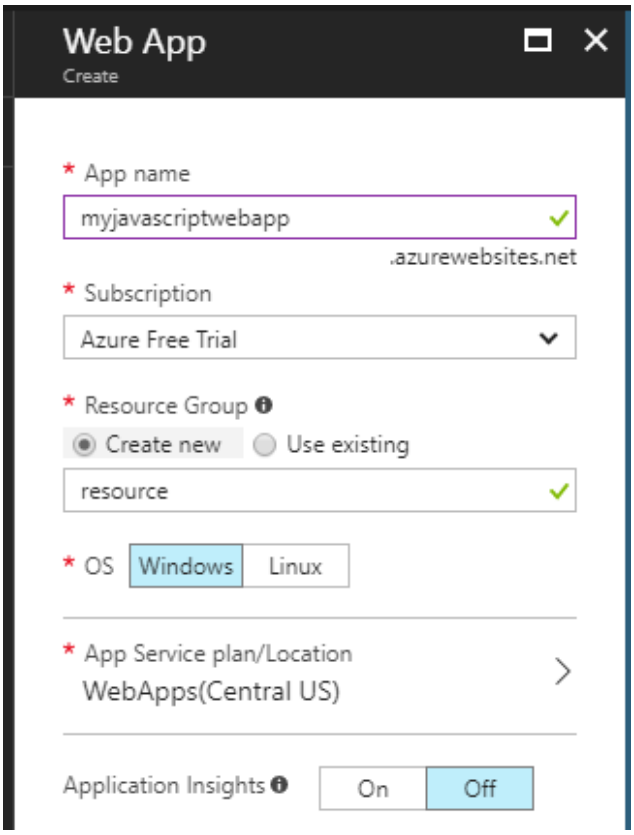
Azure DevOps je jedna ze služeb poskytovaných rozhraním Microsoft Azure. Obsahuje sadu nástrojů, které zjednodušují práci v týmu na vyvíjených projektech. Pro mě nejdůležitější nástroj byl DevOps Repos, který slouží jako privátní Git úložiště. V této službě jsem si spolu s týmem vytvořil strukturu (obrázek 7.1) a zapisoval zde řešení částí projektu. Jako návrhový podklad pro tuto strukturu sloužil ukázkový projekt, který nám byl poskytnut ze strany firmy.

7.1.2 Azure Web Sites

Azure Web Sites je služba sloužící k hostování webových aplikací. Tuto službu stačí v rozhraní Azure aktivovat a nastavit některé z požadovaných parametrů, jako je například webová adresa, na které budeme chtít aplikaci provozovat. Na obrázku 7.2 se nachází formulář, ve kterém jsou zobrazeny všechny parametry, které jsou potřeba při aktivaci nastavit (například lokace, typ předplatného Azure rozhrani atd.).



Obrázek 7.1: Struktura projektu v Azure DevOps



Obrázek 7.2: Ukázka formuláře Azure Web Sites [3]

7.2 Microsoft Visual Studio

Microsoft Visual Studio je vývojové prostředí od společnosti Microsoft. Mezi vestavěné jazyky patří C/C++, VB.NET (použitím Visual Basic .NET) a C#. Aktuální verzí je Visual Studio 2019. [4] V tomto nástroji proběhl celý vývoj webové aplikace. Jedním z důvodů využití tohoto nástroje bylo i snadné propojení se službou Azure.

7.3 Microsoft SQL Server

Microsoft SQL Server je systém správy relačních databází vyvinutý společností Microsoft. Jedná se o software produkt s primární funkcí ukládání a načítání dat - které mohou běžet na stejném počítači nebo na jiném počítači v síti (včetně Internetu). [5]

7.4 SQL Server Management Studio

SQL Server Management Studio je prostředí pro správu a vývoj databáze SQL Server. Je to komplexní nástroj, který kombinuje širokou skupinu grafických nástrojů s řadou editorů skriptů a poskytuje vývojářům a správcům databází přístup k SQL Server. [6] Toto prostředí bylo používáno především k přístupu do databáze.

7.5 ASP .NET MVC

ASP.NET MVC je webový aplikační rámec, který implementuje vzor Model-View-Controller. ASP.NET umožňuje vývojářům vytvářet webové aplikace jako složení tří komponent: modelu, pohledu a řadiče. Model obsahuje informace, které jsou zobrazovány v pohledech. Kontroler se zabývá interakcí a aktualizací modelu, aby odrážel změny ve stavu aplikace a potom předává informace do pohledu. Pohled přijímá nezbytné informace a vykresluje uživatelské rozhraní. [7] V rámci ASP .NET MVC jsem používal technologie:

7.5.1 C#

C# je vysokoúrovňový objektově orientovaný programovací jazyk vyvinutý firmou Microsoft zároveň s platformou .NET Framework. C# lze využít k tvorbě databázových programů, webových aplikací a stránek, webových služeb, formulářových aplikací ve Windows, softwaru pro mobilní zařízení a dalších. [8]

7.5.2 Razor

Razor je syntaxe značek, které umožňují do webových stránek vkládat kód založený na serveru. Razor syntaxe se skládá z Razor značek, C# a HTML. Soubory, které obsahující Razor, mají příponu .cshtml. Razor podporuje jazyk C# a používá @ symbol k přechodu z formátu HTML do C#. Razor vyhodnotí výrazy jazyka C# a generuje HTML. [9] Ukázku Razor syntaxe můžeme vidět ve výpisu 7.1.

```
@foreach(var i=0; i < 10 ; i++)
{
    <p>Number: @i</p>
}
```

Výpis 7.1: Ukázka Razor syntaxe

7.5.3 Identity

Identity neboli ASP.NET Identity je rozhraní, které podporuje funkce spravování uživatelských účtů. Právě u aplikací programovaných pomocí vzoru Model-View-Controller je toto rozhraní hojně využíváno kvůli jeho schopnosti jednoduché práce s uživatelskými účty a rolemi v celém systému. Uživatelé si mohou vytvořit účet a přihlásit se pomocí uživatelského jména a hesla nebo mohou použít externího poskytovatele přihlášení, jako je Facebook, Google, účet Microsoft, Twitter a další. [10] Funkce externího přihlášení v projektu docházkového systému nebyla zapotřebí.

7.5.4 Nager.Date

Nager.Date je oblíbené balíček rozšíření pro generování stítních svátků. V současné době podporuje více než 100 zemí. Projekt je založen na .NET. Nager.Date je software s otevřeným zdrojovým kódem a je zcela zdarma pro komerční použití. [11] Ukázku můžeme vidět ve výpisu 7.2, kde je příklad generování svátků pro stát Německo v roce 2021.

```
var publicHolidays = DateSystem.GetPublicHolidays(2021, "DE");
foreach (var publicHoliday in publicHolidays)
{
//publicHoliday.Date -> Datum
//publicHoliday.LocalName -> Lokální název svátku
//publicHoliday.Name -> Anglický název svátku
//publicHoliday.Fixed -> Informace, zda je tento svátek každý rok ve stejný den
    ...
}
```

Výpis 7.2: Ukázka Nager.Date [11]

7.6 Entity Framework

Pro přístup k databázi je využit Entity Framework, což je rámec pro tvorbu objektově-relačního mapování. Databázové tabulky se přímo mapují na C# třídy, v kódu pracujeme jen s objekty a framework sám na pozadí generuje SQL dotazy. [12] Při práci s Entity Framework jsem využil jeden z přístupů, který nese název Code-First, kdy programátor vytvoří C# třídu a podle ní se automaticky vygeneruje tabulka v databázi. Všechny dále zmíněné příkazy se zadávají v konzoli nástroje Microsoft Visual Studio. Postup při tvorbě modelu probíhá tímto zpusobem:

- Vytvoření tříd, které reprezentují tabulky v návrhu databáze.
- Deklarace vlastností, kde jsou jako typy používany vytvořené třídy, v jednom souboru zastupujícím databázový kontext. Příklad takové deklarace můžeme vidět ve výpisu 7.3.

```
public class MyContext : DbContext
{
    public DbSet<Article> Articles { get; set; }
}
```

Výpis 7.3: Ukázka deklarace vlastností [13]

- Povolení migrací příkazem **„enable-migrations“**. O migraci se dá říct, že je to popis změn mezi verzemi modelu, který má vliv na schéma v databázi. S povolením migrací v projektu vzniká i nová složka Migrations se souborem Configuration.cs, kde lze definovat v konstruktoru chování migrací. [13]
- Vytvoření nové migrace se provádí pomocí příkazu **„add-migration“** a názvu migrace. Poté, co proběhne tento příkaz se vytvoří třída, která obsahuje metody `Up` a `Down`. V těchto metodách jsou popsány změny mezi verzemi modelu. Metoda `Up` obsahuje změny, které budou přidány do databáze. Naopak metoda `Down` obsahuje změny, které budou v databázi vráceny. [13]
- Potvrzení změn se provádí příkazem **„update-database“**. Tento příkaz vygeneruje SQL skript (podle změn z třídy, která byla vytvořena předešlým příkazem) a ten se poté aplikuje do samotné databáze.

7.7 HTML

Hypertext Markup Language (zkratka HTML) je v informatice název značkovacího jazyka používaného pro tvorbu webových stránek, které jsou propojeny hypertextovými odkazy. HTML je hlavním z jazyků pro vytváření stránek v systému World Wide Web, který umožňuje publikaci dokumentů na Internetu. [14]

7.8 CSS

Kaskádové styly (v anglickém originále Cascading Style Sheets se zkratkou CSS) je v informatice jazyk pro popis způsobu zobrazení elementů na stránkách napsaných v jazycích HTML, XHTML nebo XML. [15]

7.9 Bootstrap

Bootstrap je svobodná a otevřená sada nástrojů kaskádových stylů pro tvorbu webu a webových aplikací. Obsahuje návrhářské šablony založené na HTML a CSS sloužící pro úpravu typografie, formulářů, tlačítek, navigace a dalších komponent rozhraní. Pro použití Bootstrap jsou nutné základní znalosti HTML a CSS, interaktivní prvky jako jsou tlačítka, menu a další kompletně nastavené a graficky zpracované elementy je totiž možné vložit pouze pomocí HTML a CSS. [16]

7.10 JavaScript

JavaScript je programovací jazyk, který je hojně využíván na internetových stránkách. Je to klientský skript, což znamená, že je zpracováván až v prohlížeči webové stránky. JavaScript se zapisuje přímo do HTML kódu.

7.11 DataTables

DataTables je jQuery knihovna, která zajišťuje tabulkám psaným v HTML jazyce stránkování, umožňuje jejich řádkování a spoustu dalších funkcí.

Ve výpisu 7.4 se nachází script, který HTML tabulce, která má nastavený atribut `id` na hodnotu „example“ nastaví funkce DataTables. Parametr `lenghtMenu` nastavuje možnost výběru počtu zobrazovaných záznamů v tabulce. Parametr `pagingType` říká jak budou označovány tlačítka pro stránkování. Dále parametr `emptyTable` nastavuje hlášku, která bude zobrazena, pokud v tabulce nejsou žádné záznamy.

```
<script>
  $(document).ready(function () {
    $('#example').DataTable({
      "lengthMenu": [[5, 10, 20, -1], [5, 10, 20, "All"]],
      "pagingType": 'numbers',
      "language": { "emptyTable": "No data found!" }});
  });</script>
```

Výpis 7.4: Ukázka konfigurace DataTables

7.12 Creately

Creately je vizuální nástroj s možnostmi vytváření diagramů a modelových návrhů od společnosti Cinergix. Aplikace je známá především pro vytváření vývojových diagramů. [17] Jednou z výhod nástroje je možnost snadného sdílení mezi více uživateli. Tento nástroj byl využíván k tvorbě databázového modelu.

7.13 SendGrid

SendGrid je cloudová e-mailová služba, která poskytuje zasílání e-mailů, škálovatelnost a analýzy v reálném čase spolu s rozhraními API, která usnadňují vlastní integraci. Tato služba poskytuje i vlastní balíček rozšíření v nástroji Microsoft Visual Studio. [18] Tento balíček (služba) byl využit k zajištění funkcionality automatického posílání e-mailů v projektu.

Kapitola 8

Vývoj webové aplikace

Kapitola popisuje samotný vývoj mnou tvořené webové aplikace. Při vývoji jsem vycházel z návrhu obrazovek v kapitole 6. V podkapitole 7.1.1 se nachází obrázek (7.1), který znázorňuje strukturu celého projektu docházkového systému. Struktura se skládá z několika projektů:

- DochazkovySystem.Android – Projekt, ve kterém je implementována mobilní aplikace pro operační systém Android.
- DochazkovySystem.Data – Obsahuje třídy reprezentující tabulky databáze a třídy, které byly vytvořeny technologií Entity Framework (podkapitola 7.6). Dále se zde nachází metody a funkce, které komunikují s databází.
- DochazkovySystem.FunctionApp – Projekt obsahující kód pro automaticky spouštěné aktualizace zbývajících dnů dovolené. Spouští se pomocí služby v rozhraní Azure.
- DochazkovySystem.IOS – Projekt pro mobilní aplikaci operačního systému iOS.
- DochazkovySystem.Shared – Sdílená část, kde se nachází potřebné třídy, konstanty nebo jiné části kódu, které jsou využívány v ostatních projektech. Tato část byla vytvořena za účelem toho, aby nedocházelo k duplikaci kódu.
- DochazkovySystem.Web – Projekt mnou vyvíjené webové aplikace, která je detailněji popsána v dalších podkapitolách.
- DochazkovySystem.WebApi – Projekt pro webové rozhraní API, které poskytuje mobilním aplikacím potřebná data.

8.1 Registrace, přihlášení a přístupové role

Registrace a přihlášení byl můj první krok při programování webové aplikace. Programování probíhalo pomocí technologie ASP .NET Identity (podkapitola 7.5.3), která se stará právě o uživatelské

účty a přidělování rolí. Po nasazení těchto funkcionalit stačí použít pro ověření uživatelské role funkci `IsInRole`.

```
if (User.IsInRole("COMPANY-ADMIN"))
{
   //kód pro přesměrování uživatele na danou stránku
}
```

Výpis 8.1: Ukázka funkce k ověření role uživatele

Samotná registrace probíhá následovně. Administrátor vytvoří nového zaměstnance (popsáno v podkapitole 8.2). Na zadaný email zaměstnance je zaslán odkaz na formulář, kde registrace pokračuje. Základ tohoto formuláře byl vytvořen již zmiňovanou technologií ASP. NET Identity, ale musel být přetvořen pomocí CSS stylu, aby se shodoval s grafickým návrhem ostatních stránek. Ve formuláři uživatel vyplní email a dvakrát nové heslo pro kontrolu. Pokud je vše v pořádku, může se přihlásit do systému. V opačném případě (například zadá špatný email) dostane upozornění v podobě chybové hlášky.

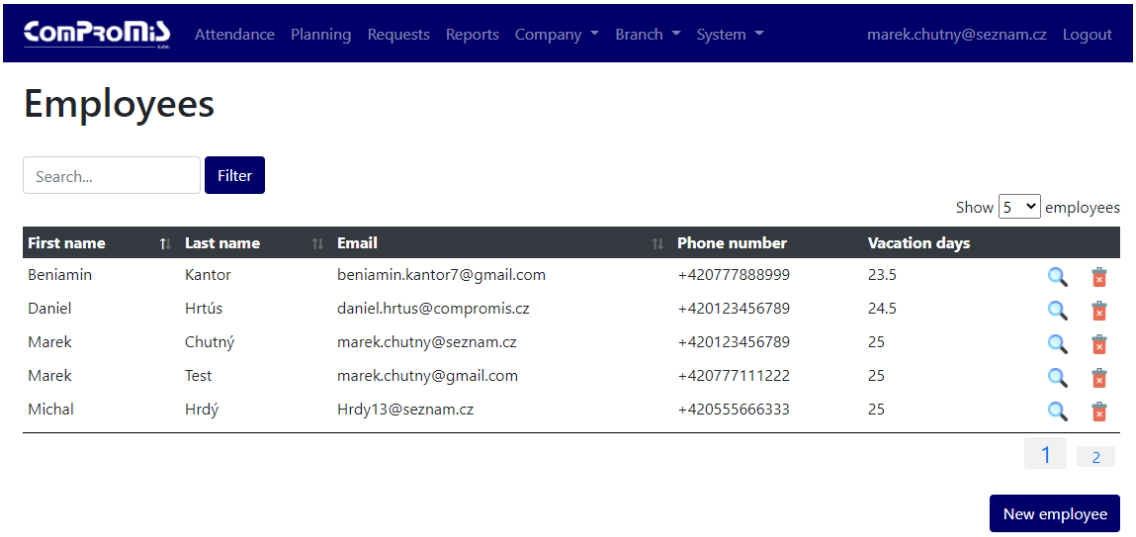
Ve výpisu 8.2 se nachází metody, která má za úkol posílaní e-mailů. Metoda `configSendGridasync` očekává jako vstupní parametr objekt třídy `IdentityMessage`, která je také automaticky generována technologií ASP .NET Identity (podkapitola 7.5.3). Tento objekt v sobě nese informace, které jsou potřeba k odeslání e-mailu (předmět, zprávu, email korespondenta atd.). V této metodě bylo zapotřeba se pripojit ke službě, která se bude o zasílaní automatických emailů starat, kolegy z firmy mi byla doporučena služba SendGrid (podkapitola 7.13). V této službě jsem provedl registraci, kdy mi byl vygenerován API klíč, díky kterému se lze k službě vzdáleně připojit. Poté stačí jen vytvořit a naplnit obejkt třídy `SendGridMessage`, která je součástí balíčku služby. Poté je poslán požadavek službě SendGrid, která pošle e-mail, dle vlastností posílaného objektu třídy `SendGridMessage`.

```
    private async Task configSendGridasync(IdentityMessage message)
    {
      var apiKey = "...";
      var client = new SendGridClient(apiKey);
      var msg = new SendGridMessage();
      msg.AddTo(message.Destination);
      msg.From = new EmailAddress("chu0077@vsb.cz", "DochazkovySystem");
      msg.Subject = message.Subject;
      msg.PlainTextContent = message.Body;
      msg.HtmlContent = message.Body;
      var response = await client.SendEmailAsync(msg);
    }
```

Výpis 8.2: Ukázka posílání emailů

8.2 Stránky Employees, Branches a Types of employment

Dále bylo potřeba naprogramovat stránky, kde administrátor své pobočky (firmy) bude moct spravovat data (například vytvářet zaměstnance, typy úvazků). Všechny tyto stránky jsou založeny na podobném principu. Princip představím na stránce Employees (obrázek 8.1).



Obrázek 8.1: Ukázka stránky „Employees“

V horní části je textové pole, který slouží jako filtr. Po zadání textového řetězce do tohoto pole a kliknutí na tlačítko „Filter“ se zobrazí pouze data obsahující tento řetězec ve jméně nebo emailu. Dále se zde nachází HTML tabulka a s pomocí tehnologie DataTables (podkapitola 7.11) se zobrazuje seznam zaměstnanců dané pobočky či firmy. V dolní části je tlačítko pro přidání zaměstnance, kdy po jeho kliknutí se zobrazí formulář s požadovanými informacemi jako je email, jméno, role, typ úvazku a tak dále. Po vyplnění a odeslání formuláře se uživatel vloží do databáze v případě, že jsou všechny informace správně vyplněny. Pokud správně vyplněny nejsou, zobrazí se chybová hláška. Zaměstnanci jsou automaticky přiřazeny plány státních svátků a plány pobočky či firmy. Na této stránce je i funkcionalita možnosti rozkliknutí detailu zaměstnance, kdy se zobrazí formulář, který je totožný s formulářem pro přidání zaměstnance s tím rozdílem, že tento formulář je naplněn daty z databáze, které mohou být zaměstnanci upraveny. V poslední řadě je zde tlačítko pro smazání.

Výpis 8.3 ukazuje metodu, která se volá při odeslání formuláře. Pro přenos potřebných dat z formuláře se nejčastěji používá požadavek typu `POST`. Metoda, která má být použita pro zpracování tohoto typu požadavku, je označena v hranatých závorkách atributem `HttpPost`. Ve výpisu 8.3 je ukázka metody pro zpracování požadavku typu `POST`, která jako parametr očekává objekt reprezentující formulář, ve kterém jsou všechna potřebná data. Poté je volaná metoda `InsertWithPlans` třídy `userService`, což je třída, která se nachází v projektu `DochazkovySystem.Data` (7.1).

```
[HttpPost]
public async Task<ActionResult> AddEmployee(UserForm u)
{
  if (ModelState.IsValid)
     {
         var userserviceResult = await userService.InsertWithPlans(u.FirstName, u.
            LastName, u.Email, u.PhoneNumber, "", u.Branch.VacationDays, 0, u.
            Branch.Id,u.WorkPlanId);
        ...
    }
}
```

Výpis 8.3: Ukázka metody pro přidání zaměstnance

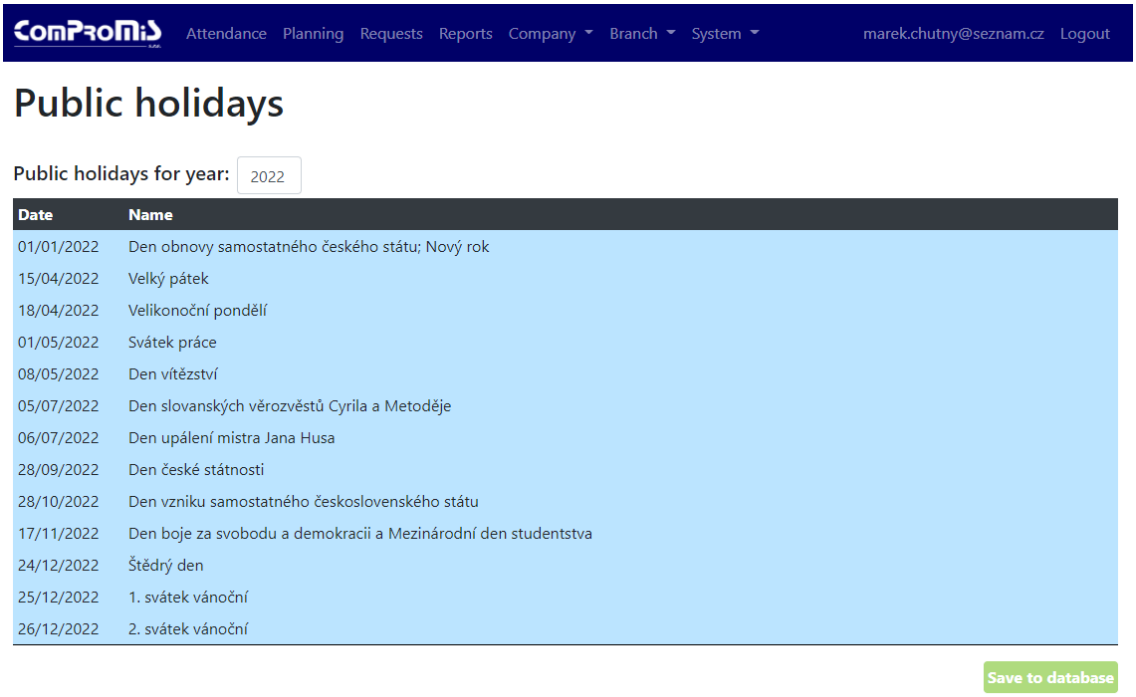
Výpis 8.4 ukazuje část metody „InsertWithPlans“ pro vytvoření nových záznamů v databázi. Stačí vytvořit nový objekt typu `User`, který je součástí ORM, a nastavit jeho parametry údaji zadanými ve formuláři. Poté se zaměstnanec přidá do kolekce pomocí funkce `Add`. Dále se pomocí LINQ dotazu vyberou všechny pracovní plány, kde je atribut `Priority` nastaven na hodnotu 1 nebo 2. Tyto plány jsou procházeny cyklem, kdy při každé iteraci je vytvořen objekt typu `UserWorkPlan`, kde je do parametru `User` přiřazen objekt nového zaměstnance (`u`) a do parametru `WorkPlan` je přiřazen objekt plánu aktuální iterace. Poté je objekt typu `UserWorkPlan` vložen pomocí funkce `Add` do kolekce `UserWorkPlans` . Po ukončení cyklu jsou všechny provedené změny potvzeny příkazem `SaveChangesAsync`.

```
 Branch b = db.Branches.Find(BranchId);
 User u = new User
  {
     Id = Guid.NewGuid().ToString(),
     FirstName = FirstName,
     LastName = LastName,
     Email = Email,
     PhoneNumber = PhoneNumber,
     OwnVacationDays = OwnVacationDays,
     CompensatoryTimeOff = CompensatoryTimeOff,
     Branch = b,
     UserName = Email
  };

db.Users.Add(u);

var PublicAndBranchHolidays = db.WorkPlans.
Where(x => x.Branch.Id == b.Id).Where(x => x.Priority == 1 || x.Priority == 2);

     foreach (var holiday in PublicAndBranchHolidays)
     {
        UserWorkPlan uw = new UserWorkPlan
        {
           User = u,
           WorkPlan = holiday,
        };

     db.UserWorkPlans.Add(uw);

     }

await db.SaveChangesAsync();
```

Výpis 8.4: Ukázka obsahu metody „InsertWithPlans“

8.3 Stránka Public holidays

Tato stránka je určena zejména pro administrátora celého systému.



Obrázek 8.2: Ukázka stránky „Public holidays“

V horní části (obrázek 8.2) se nachází textové pole, do které se zadává rok, pro který chceme svátky generovat. Ve výchozím stavu je tento rok nastaven na rok aktuální. Při tvorbě této funkcionality stránky jsem využil Nager.Date (podkapitola 7.5.4). Pokud uživatel zadá rok, automaticky se odešle pomocí JavaScriptu požadavek na výpis těchto státní svátků. Svátky se pro daný rok vygenerují. Poté se pomocí LINQ dotazu vyberou svátky pro daný rok z databáze, kde jsou uloženy ve formě pracovních plánů. Pokud počet vygenerovaných svátků souhlasí s počtem svátků v databázi, vypíšou se do HTML tabulky. V opačném případě se zobrazí upozornění, že svátky pro daný rok neexistují a ukáže se tlačítko „Generate public holidays“. Po kliknutí se znovu vygenerují svátky, které se pomocí iterací v cyklu vkládají do modelové třídy stránky a vypíšou do tabulky pomocí tagů jazyka HTML. V dolní části se zobrazí nové tlačítko „Save to databse“, které vytvoří pracovní plány a přiřadí je všem uživatelům systému.

Ve výpisu 8.5 se nachází třída `PublicHolidaysView`, která je použita jako model stránky. Třída `PublicHoliday` označuje státní svátky spolu s kalendářním datem konání a jejich názvem.

```
public class PublicHolidaysView
{
    public int? Year { get; set; }
    public List<PublicHoliday> Holidays { get; set; }
}

public class PublicHoliday
{
    public DateTime Date { get; set; }
    public string Name { get; set; }
}
```

Výpis 8.5: Ukázka modelů stránky

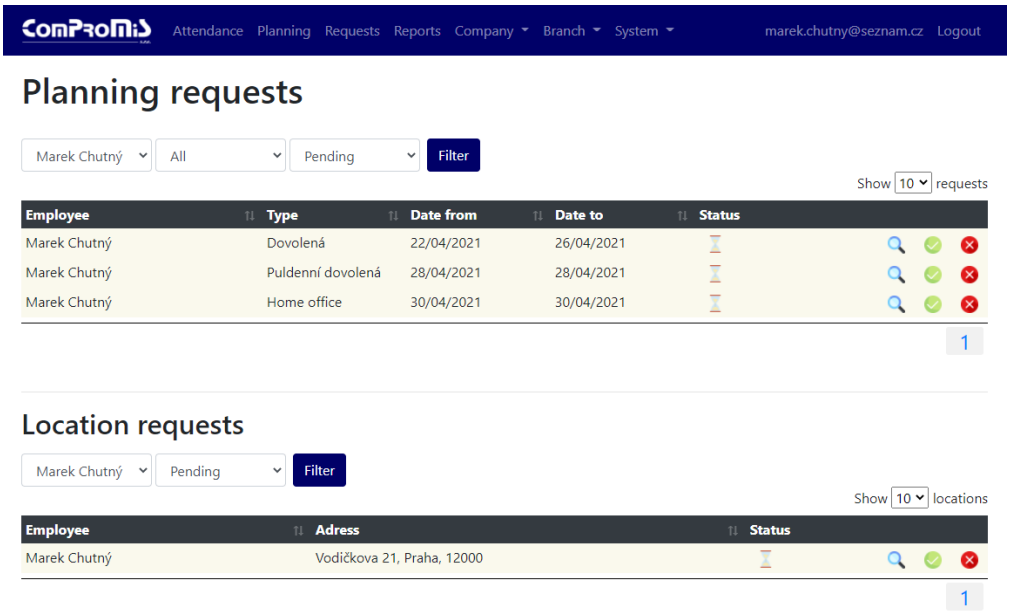
Výpis 8.6 ukazuje generování svátků, kdy Nager.Date pomocí zadaného roku a kódu dané země vrací státní svátky. Tyto svátky poté prochází cyklus, který v každé iteraci vytváří objekt třídy `PublicHoliday`, ten je poté vložen do kolektce třídy `PublicHoliday`. Nakonec je tato kolekce přiřazena do modelu zobrazení.

```
PublicHolidaysView model=new PublicHolidaysView { Year=DateTime.Now.Year };
var publicHolidays = DateSystem.GetPublicHoliday(DateTime.Now.Year, "CZ");
var holidays = new List<PublicHoliday>();
foreach (var publicHoliday in publicHolidays)
       {
           holidays.Add(new PublicHoliday { Date = publicHoliday.Date, Name =
               publicHoliday.LocalName });
       }
model.Holidays=holidays;
```

Výpis 8.6: Ukázka generování svátku pro daný rok

8.4 Stránka Requests

Dalším krokem byla tvorba stránky, kde budou vyobrazeny všechny žádosti zaměstnanců pobočky. Stránka je rozdělená do dvou částí (obrázek 8.3).



Obrázek 8.3: Ukázka stránky „Requests“

První část (tzn. Planning requests) slouží k požadavkům na vytvoření nových plánů zaměstnanců. Nachází se zde tři rozbalovací seznamy, které slouží jako filtry (zaměstnance, typu požadavku a statusu). Poslední rozbalovací seznam je nastaven na výchozí hodnotu „Pending“, která představuje pouze žádosti, čekající na scvhálení. Pod těmito filtry se pak nachází tabulka tvořena HTML tagy a pomocí technologie DataTables nám tyto žádosti vyobrazuje. Na každém řádku se nachází tlačítka pro zobrazení detailu, schválení a zamítnutí požadavku.

Druhá část slouží jen k požadavkům ke schválení nového místa, které může zaměstnanec využívat jako home office. Funkčnost a zobrazení je zde stejné jako u první části s tím rozdílem, že místo tří rozbalovacích seznamů se zde nachází pouze dva pro filtr zaměstnance a statusu schválení (nevybíráme zde typ požadavků).

Nejnáročnější fáze při programování této stránky, byla situace po schválení požadavků. Po schválení požadavků se automaticky vygeneruje nebo nevygeneruje pracovní plán. Každý plán je tvořen rozpadem (například, když uživatel požádá o třídenní dovolenou, budou vytvořeny tři plány, pro každý den jeden, pokud je to možné). Cyklus prochází dny, které uživatel zadal při tvorbě požadavku. Následuje podmínka, kdy je kontrolováno, zda procházený den není víkend (výpis 8.7), protože plány o víkendu není možné ve firmě tvořit. Pokud ano, je tento den ignorován a cyklus přejde na další iteraci.

```
if (day.DayOfWeek == DayOfWeek.Saturday || day.DayOfWeek == DayOfWeek.Sunday)
{ continue; }
```

Výpis 8.7: Ukázka kontroly víkendu

Pro dny, kdy se nejedná o víkend, jsou pomocí LINQ dotazu vybrány všechny plány pro tento den. Tyto plány jsou procházeny v cyklu, ve výpisu 8.8 jsou ukázany podmínky tohoto cyklu. Pokud má právě procházený plán nastavený atribut `Priority` na hodnotu 1 nebo 2 a zároveň atribut `Type` je „Absence“ plán se na daný den nevytvoří (protože v dny, kdy je plánován svátek nebo celofiremní dovolená uživateli, nemůže být vytvořen). V další podmínce je vyřešena situace, kdy si uživatel chce vzít půldenní dovolenou, ale zároveň má v daný den již vytvořen plán s nastaveným atributem `Type` na hodnotu „Productive“ (například home office) nebo naopak, a zároveň na daný den nesmí mít vytvořené dva plány. Pokud tato podmínka není splněna, jsou všechny plány pro tento den procházeny a poté kontrolovány, zda plán v iteraci má atribut `Priority` hodnotu 2, pokud je podmínka splněna není mazán plán, ale pouze záznam z vazební tabulky `UserWorkPlans`. Dále je kontrolováno, zda plán v iteraci nemá nastavený atribut `Priority` na 1 (státní svátek), pokud je podmínka splněna je mazán plán z tabulky `WorkPlans`.

```
if ((plan.WorkPlan.Priority == 1 || (plan.WorkPlan.Priority == 2 && plan.WorkPlan.
    ActivityType.Type == ActionType.Absence)))
{ break; }
if (!((req.Type.Type == ActionType.Productive && plan.WorkPlan.ActivityType.Code
    == "HDH") || (plan.WorkPlan.ActivityType.Type == ActionType.Productive && req.
    Type.Code == "HDH")) && ThisDayPlans.Count < 2))
{ foreach (var delplan in ThisDayPlans)
      {
         if (delplan.WorkPlan.Priority == 2) {
           var deleteuserfromplan = db.UserWorkPlans.Where(x => x.WorkPlan.Id ==
               delplan.WorkPlanId && x.UserId ==req.User.Id).FirstOrDefault();
           db.UserWorkPlans.Remove(deleteuserfromplan); }
         else if (delplan.WorkPlan.Priority != 1) {
           db.WorkPlans.Remove(db.WorkPlans.Find(delplan.WorkPlanId));}
       } break;
}
```

Výpis 8.8: Ukázka podmínek při tvoření plánu

Ve výpisu 8.9 se nachází podmínka pro situaci, kdy administrátor pobočky chce vytvořit celozávodní plán, ale zaměstnanec má přiřazen již jiný pracovní plán. Podmínka je splněna pokud má zaměstnanec plánovanou nemocenskou (atribut `Code` je roven hodnotě „S“) nebo má plán, kdy nemá být v práci (atribut `Type` je roven hodnoty „Absence“) a zároveň plán pobočky je pracovního typu (atribut `Type` je roven hodnoty „Productive“). Jestliže je podmínka splněna zaměstnanci nebude přiřazen plán pobočky.

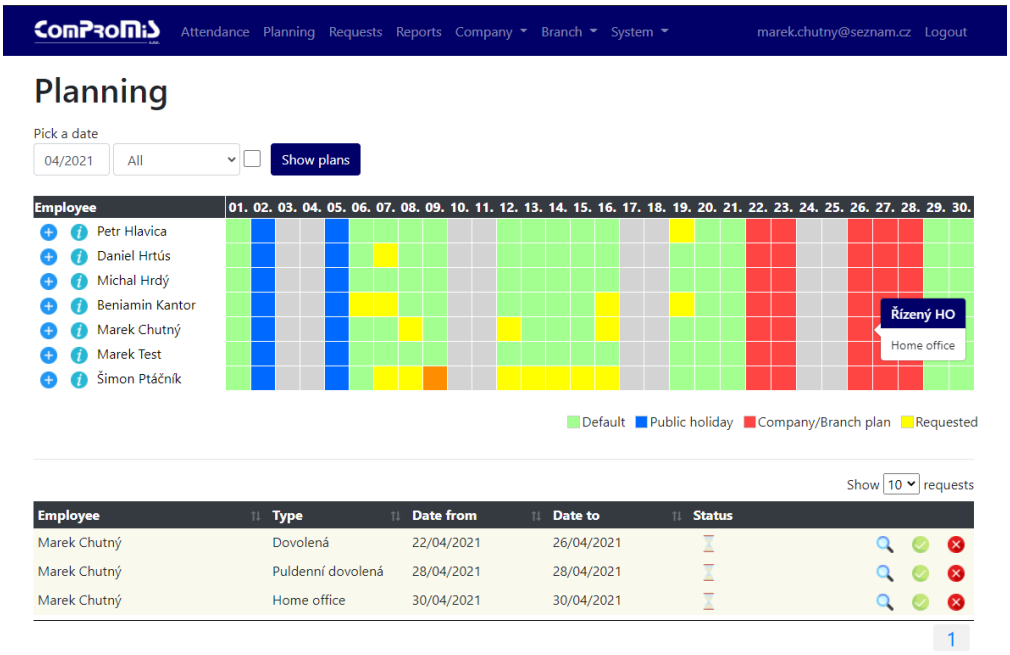
```
if (UserDayPlan.WorkPlan.ActivityType.Code == "S" ||((UserDayPlan.WorkPlan.
    ActivityType.Type==ActionType.Absence && UserDayPlan.WorkPlan.Priority==3) &&
    type.Type==ActionType.Productive))
 { continue; }
```

Výpis 8.9: Ukázka podmínek při tvoření celozávodního plánu

Pokud podmínka není splněna jsou všechny plány zaměstnance pro daný den smazány a zaměstnanci je přiřazen plán pobočky.

8.5 Stránka Planning

Tato obrazovka byla při programování jedna z těch složitějších. Stejně jako stránka Requests je rozdělena do dvou částí.



Obrázek 8.4: Ukázka stránky „Planning“

První část se používá ke sledování zaměstnaneckých plánů, jejich mazání a přidávání. Nachází se zde textové pole, do kterého se zadává měsíc a rok ve formátu (MM/YYYY), ve výchozím stavu je nastaveno na aktuální datum. Vedle se nachází rozbalovací seznam a zaškrtávací pole. Textové pole slouží k filtrování zaměstnance, pro kterého chceme ukázat plány. Pokud je zaškrtavací pole aktivní, skryjí se víkendy a zobrazí se pouze všední dny v tabulce, která je popsána dále. V prvním sloupci je pro každého zaměstnance pobočky tlačítko, které zobrazí modální okno s možností přidání

nového plánu a tlačítko s možnosti ukázání všech plánů a jejich smazání. Další sloupce tabulky pak prezentují určitý den v měsíci. U této části bylo velice důležité kontrolovat při přidávání plánu, zda už na daný den plán neexistuje. Náročné u těchto plánů bylo i uvědomit si, v které dny plán může být vytvořen nebo za jakých podmínek může na daný den existovat více plánů. Při vyobrazení plánu musel byl hledán vždy plán s nejvyšší prioritou a podle této priority pak musela být buňka tabulky zabarvená pomocí CSS stylu.

Druhá část obsahuje tabulku s žádostmi ve statusu „Pending“ s možností jejich sválení, zamítnutí a zobrazení detailů podobně jako na stránce Requests.

Výpis 8.10 popisuje zadávání nového data do textového pole. Je důležité si povšimnout scriptového kódu. Za pomocí JavaScript knihovny Bootstrap-Datepicker[1] je zajištěno, že při kliknutí na textové pole, které má nastaveno atribut `id` na „datepicker“, se ukáže kalendář ve formátu, který je nastaven pomocí vlastnosti `format` (v tomto případě ve tvaru "mm/yyyy"). Další funkce v scriptu zajišťuje, že při kliknutí na požadovaný měsíc v kalendáři se získají hodnoty z tohoto kalendáře. Získaný měsíc se nastaví jako hodnota prvku, jehož atribut `id` je nastaven na „Month“ a získaný rok je nastaven jako hodnota prvku, jehož atribut `id` je nastaven na „Year“. Tyto prvky určují vlastnosti modelu třídy této stránky a jsou tvořeny pomocí Razor syntaxe. Tento způsob zajišťuje, že vlastnosti modelu třídy se nezmění, dokud uživatel nevybere hodnotu z kalendáře.

```
<input type="text" class="form-control float-left mr-1" name="datepicker" id="
    datepicker" placeholder="mm/yyyy" data-val-required="Date is required."
    data-val="true" autocomplete="off"/>
@Html.HiddenFor(m => m.Year, new { @id = "Year" })
@Html.HiddenFor(m => m.Month, new { @id = "Month" })
<script>
    var dp = $("#datepicker").datepicker({
        format: "mm/yyyy",
        startView: "months",
        minViewMode: "months",
        orientation: 'bottom'
    });
    dp.on('changeDate', function (e) {
        var pickedMonth = new Date(e.date).getMonth() + 1;
        var pickedYear = new Date(e.date).getFullYear();
        document.getElementById('Month').value = pickedMonth;
        document.getElementById('Year').value = pickedYear;
    }); </script>
```

Výpis 8.10: Ukázka kódu pro změnu datumu

[1] https://bootstrap-datepicker.readthedocs.io/en/latest/

Ve výpisu 8.11 je ukázán LINQ dotaz pro výběr plánu na daný měsíc (`Month`) a rok (`Year`) pro uživatele podle jeho identifikačního čísla (`userId`).

```
var query =await db.UserWorkPlans.Where(x => x.User.Id == userId)
                        .Where(x =>x.WorkPlan.StartDate.Value.Year <= Year)
                        .Where(x => x.WorkPlan.EndDate.Value.Year >= Year)
                        .Where(x => x.WorkPlan.StartDate.Value.Month <= Month)
                        .Where(x => x.WorkPlan.EndDate.Value.Month >= Month)
                        .OrderBy(x=>x.WorkPlan.Priority)
                        .ToListAsync();
```

Výpis 8.11: Ukázka výběru plánu

Výpis 8.12 obsahuje cyklus, který prochází pracovní plány získáné z předešlého LINQ dotazu (8.11), kdy cyklus hledá plán s nejvyšší prioritou pro daný den. Proměnná `prio` slouží pro ukládání nejvyšší priority z procházených plánů. Proměnná `plansum` slouží jako počítadlo plánů, protože zaměstnanec může mít více pracovních plánů v jeden den (například půldenní dovolenou a home office).

```
int prio=0;
int plansum=0;
foreach (var checkplan in Data.workPlans)
{
  if (checkplan.WorkPlan.Priority >= prio && checkplan.WorkPlan.StartDate <= date
      && date <= checkplan.WorkPlan.EndDate)
      {
        if(checkplan.WorkPlan.Priority>=2)
        { plansum++; }
        prio = checkplan.WorkPlan.Priority;
      }
}
```

Výpis 8.12: Ukázka počítání plánu s nejvyšší prioritou

8.6 Stránka Attendance

V předposlední fázi jsem programoval tu nejdůležitější obrazovku tohoto systému, která slouží k administraci všech aktivit zaměstnance, jejich vyobrazování na časové ose a schvalování. Obrazovka má nahoře funkcionalitu přepínání do denního režimu, kde se zobrazí aktivity všech zaměstnanců na určitý den. Hlavní důraz byl kladen na funkčnost stránky pro týdenní přehled těchto aktivit. Dominantou stránky je velká tabulka. V prvním sloupci tabulka obsahuje tlačítko, které otevírá

modální okno s možností přidání aktivity. V druhém sloupci datum dne. Třetí počítá odpracované hodiny z proběhlých aktivit. Ve čtvrém sloupci je počítán přesčas. Dále v tabulce také najdeme ikonku, která zobrazuje plán zaměstnance na daný den. Zbytek tabulky zobrazuje čas.

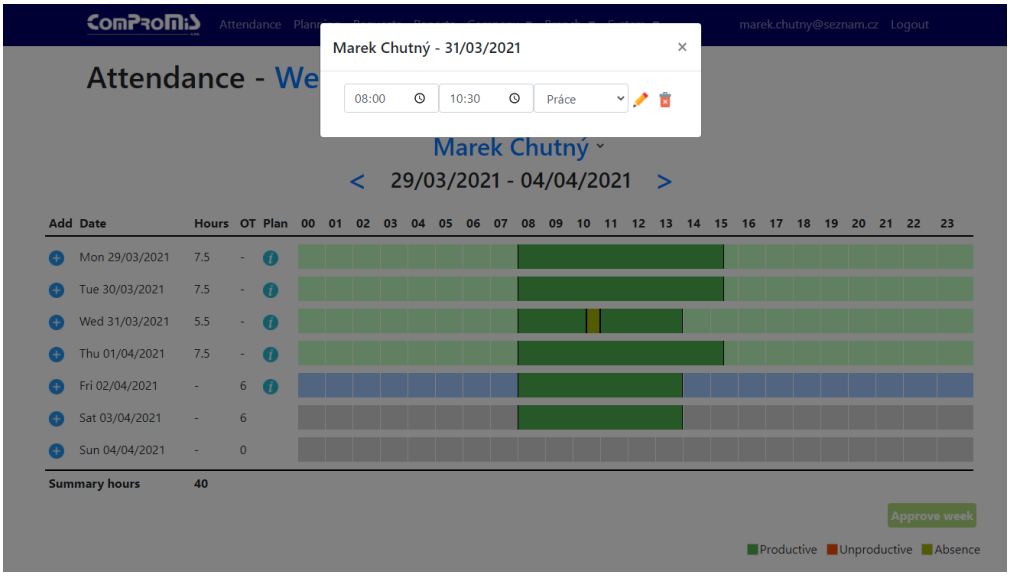
Časová osa je tvořena buňkami HTML tabulky. Každý taková buňka reprezentuje 5 minut dne. Každý den je vyzobrazen 288-mi buňkami [(24 hodin * 60 minut) / 5)]. Buňky jsou zabarveny podle typu plánu na den pomocí CSS stylu.

Do zobrazení se posílají ukončené aktivity zaměstnance seřazené podle atributu `Start` pomocí LINQ dotazu, který je zobrazen ve výpise 8.13.

```
var activities = await db.Activities.Include("Type").AsQueryable()
.Where(x => x.User.Id == userId)
.Where(x => x.Start.Value >= startOfWeek.Date && x.End.Value <= endOfWeek)
.OrderBy(x => x.Start).Where(x => x.End.Value!=null).ToListAsync();
```

Výpis 8.13: Ukázka LINQ dotazu pro ukončené aktivity za týden

Tyto aktivity jsou ve zobrazení procházený cyklem, který kontroluje, zda aktivita probíhala v daný den. V každé iteraci, kdy je podmínka splněna, je začátek a konec aktivity převáděn na minuty, tyto minuty jsou potom vyděleny 5-ti. Tímto je získán interval buněk, které má aktivita zaplňovat. Tento interval je pak procházen a pomocí CSS stylu vybarven sytou barvou dle typu aktivity.



Obrázek 8.5: Ukázka formuláře pro úpravu aktivity

Po kliknutí na aktivitu se zobrazuje modální okno s časem a typem aktivity, kdy administrátor může aktivitu smazat nebo upravit (obrázek 8.5). U upravování a přidávání aktivity bylo třeba také naprogramovat kontrolu, zda tato aktivita nezasahuje do jiné. Spravování aktivit je povoleno pouze

pro aktivity již proběhlých dní. Pokud se administrátor snaží upravit, smazat či přidat aktivitu do dne, který ještě neproběhl je upozorněn chybovou hláškou.

Ve výpisech 8.14 a 8.15 je ukázka řešení situace, kdy administrátor pobočky, chce přidat pomocí formuláře novou aktivitu danému uživateli. Výpis 8.14 ukazuje LINQ dotaz, kterým získávám aktivity pro daného uživatele v daný den.

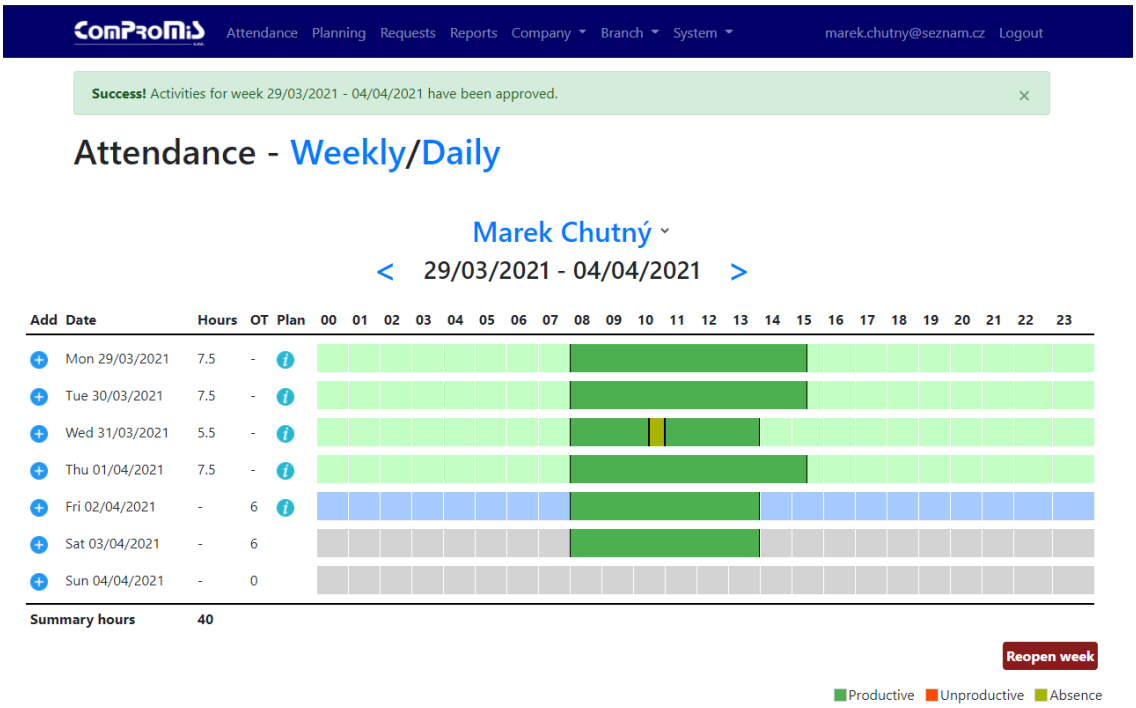
```
var activities = await db.Activities.Include("Type").AsQueryable()
.Where(x => x.User.Id == userId)
.Where(x => x.Start.Value.Year==day.Year && x.Start.Value.Month==day.Month &&
        x.Start.Value.Day==day.Day)
.OrderBy(x => x.Start)
.ToListAsync();
```

Výpis 8.14: Ukázka získaní aktivit z databáze

Ve výpisu 8.15 se nachází ukázka kontroly překryvu nové aktivity s ostatními aktivitami pro daný den uživatele. Jsou zde používány aktivity, které byly získány pomocí dotazu z výpisu 8.14. V cyklu jsou tyto aktivity procházeny. U každé aktivity v iteraci je kontrolováno, zda má nastaven atribut `StartChanged`, pokud ano, do proměnné `s` uložím jeho hodnotu. V opačném případě je uložena hodnota atributu `Start`. Stejný proces funguje i pro atribut `EndChanged`. Dále je proměnná `s` a `e` porovnávána s hodnotami časových údajů získaných z formuláře pro přidání nové aktivity.

```
if (activity.Start.Value.Date == form.Day)
{
DateTime s = activity.StartChanged != null ? (DateTime)activity.Start : (DateTime)
    activity.StartChanged;
DateTime e = activity.EndChanged != null ? (DateTime)activity.End : (DateTime)
    activity.EndChanged;

    if (((s.TimeOfDay < form.Start.TimeOfDay) &&
    (form.End.TimeOfDay < e.TimeOfDay)) || ((form.Start.TimeOfDay < s.TimeOfDay)
        && (form.End.TimeOfDay > s.TimeOfDay)) ||
    ((form.Start.TimeOfDay > s.TimeOfDay) &&
    (form.Start.TimeOfDay < e.TimeOfDay)))
    {
        //přesměrování na stránku
    }
}
```

Výpis 8.15: Ukázka testování nové aktivity



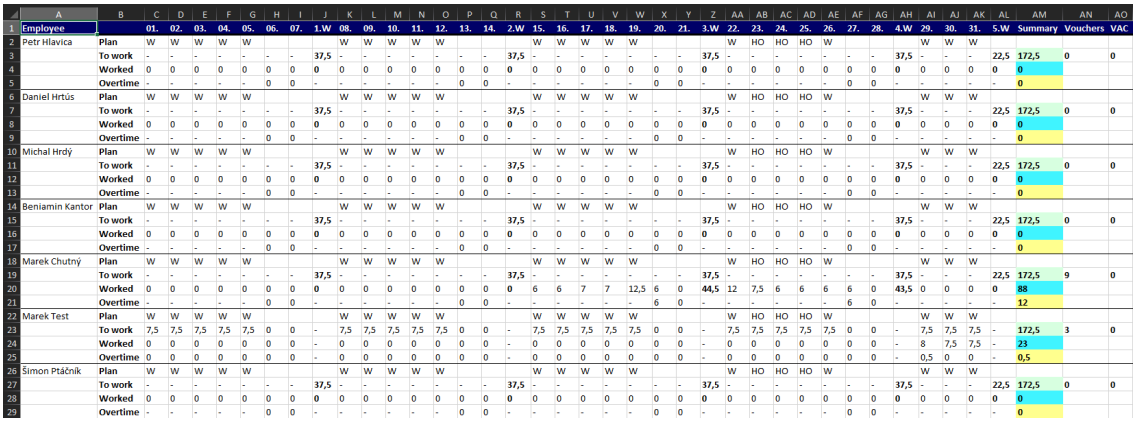
Obrázek 8.6: Ukázka stránky „Attendance“

Schválení všech aktivit umožňuje tlačítko „Approve week“. Kdy je zaměstnanci kontrolován typ úvazku, probíhá kontrola odpracovaných hodin a kontrola hodin, které měl odpracovat za daný časový úsek. Další podmínkou pro schválení je ověření povinného pracovního času, kdy zaměstnanec musí pracovat. Pokud kontrola proběhne úspěšně, tak jsou aktivity schváleny a tlačítko se změní na „Reopen week“ (obrázek 8.6). Vzhledem k tomu, že schvalování probíhá týdně, stačí zjistit status schválení první aktivity, pokud nemá hodnotu „Pending“ počítá se týden jako schválený. V tento čas administrátor již nemůže aktivity spravovat. Pokud potřebuje v daném týdnu pro zaměstnance provést změnu, klikne na tlačítko „Reopen week“, aktivity již nebudou ve schváleném stavu a administrátor může znova spravovat aktivity. Pokud zaměstnanec neodpracoval požadovaný počet hodin nebo nesplnil povinnou pracovní dobu, zobrazí se chybová hláška.

8.7 Stránka Reports

Jedna z posledních fází vývoje byla věnována naprogramování přehledové tabulky. Stránka obsahuje textové pole, do kterého se zadává měsíc a rok ve formátu (MM/YYYY). Měsíc i rok musí být menší nebo rovno dnešnímu datu. Pro měsíc a rok, který by byl větší než ten aktuální, tato funkce ztrací význam, protože žaměstnanci nevykonali žádné aktivity. Po kliknutí na tlačítko „Generate report“ se nám zobrazí HTML tabulka, která vyobrazuje pro každého zaměstnance, plán na všechny dny měsíce a také informace jako jsou, kolik hodin měl odpracovat, jak dlouho pracoval nebo počet

přesčasů. Tyto informace jsou počítány ze schválených aktivit. Dále jsem zde musel napsat kód programu i pro výpočet stravenek, na které má zaměstnanec nárok nebo počet dovolené čerpáné během zvoleného měsíce. Součástí této stránky je tlačítko „Export to excel“. Pro tuto funkci jsem využil knihovnu s názvem EPPlus[2]. Tato knihovna obsahuje třídu `ExcelPackage`, která umožňuje tvořit .xlsx soubory (obrázek 8.7).



Employee		01.	02.	03.	04.	05.	06.	07.	1.W	08.	09.	10.	11.	12.	13.	14.	2.W	15.	16.	17.	18.	19.	20.	21.	3.W	22.	23.	24.	25.	26.	27.	28.	4.W	29.	30.	31.	5.W	Summary	Vouchers	VAC
Petr Hlavica	Plan	W	W	W	W	W				W	W	W	W	W				W	W	W	W	W				W	HO	HO	HO	W				W	W	W				
	To work	-	-	-	-	-	-	-	37,5	-	-	-	-	-	-	-	37,5	-	-	-	-	-	-	-	37,5	-	-	-	-	-	-	-	37,5	-	-	-	22,5	172,5	0	0
	Worked	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
	Overtime	-	-	-	-	-	0	0	-	-	-	-	-	-	-	0	0	-	-	-	-	-	0	0	-	-	-	-	-	-	0	0	-	-	-	-	-	0		
Daniel Hrtús	Plan	W	W	W	W	W				W	W	W	W	W				W	W	W	W	W				W	HO	HO	HO	W				W	W	W				
	To work	-	-	-	-	-	-	-	37,5	-	-	-	-	-	-	-	37,5	-	-	-	-	-	-	-	37,5	-	-	-	-	-	-	-	37,5	-	-	-	22,5	172,5	0	0
	Worked	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
	Overtime	-	-	-	-	-	0	0	-	-	-	-	-	-	-	0	0	-	-	-	-	-	0	0	-	-	-	-	-	-	0	0	-	-	-	-	-	0		
Michal Hrdý	Plan	W	W	W	W	W				W	W	W	W	W				W	W	W	W	W				W	HO	HO	HO	W				W	W	W				
	To work	-	-	-	-	-	-	-	37,5	-	-	-	-	-	-	-	37,5	-	-	-	-	-	-	-	37,5	-	-	-	-	-	-	-	37,5	-	-	-	22,5	172,5	0	0
	Worked	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
	Overtime	-	-	-	-	-	0	0	-	-	-	-	-	-	-	0	0	-	-	-	-	-	0	0	-	-	-	-	-	-	0	0	-	-	-	-	-	0		
Beniamin Kantor	Plan	W	W	W	W	W				W	W	W	W	W				W	W	W	W	W				W	HO	HO	HO	W				W	W	W				
	To work	-	-	-	-	-	-	-	37,5	-	-	-	-	-	-	-	37,5	-	-	-	-	-	-	-	37,5	-	-	-	-	-	-	-	37,5	-	-	-	22,5	172,5	0	0
	Worked	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
	Overtime	-	-	-	-	-	0	0	-	-	-	-	-	-	-	0	0	-	-	-	-	-	0	0	-	-	-	-	-	-	0	0	-	-	-	-	-	0		
Marek Chutný	Plan	W	W	W	W	W				W	W	W	W	W				W	W	W	W	W				W	HO	HO	HO	W				W	W	W				
	To work	-	-	-	-	-	-	-	37,5	-	-	-	-	-	-	-	37,5	-	-	-	-	-	-	-	37,5	-	-	-	-	-	-	-	37,5	-	-	-	22,5	172,5	9	0
	Worked	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	6	7	7	12,5	6	0	44,5	12	7,5	6	6	6	6	0	43,5	0	0	0	0	88		
	Overtime	-	-	-	-	-	0	0	-	-	-	-	-	-	0	0	-	-	-	-	-	-	6	0	-	-	-	-	-	-	6	0	-	-	-	-	-	12		
Marek Test	Plan	W	W	W	W	W				W	W	W	W	W				W	W	W	W	W				W	HO	HO	HO	W				W	W	W				
	To work	7,5	7,5	7,5	7,5	7,5	0	0	-	7,5	7,5	7,5	7,5	7,5	0	0	-	7,5	7,5	7,5	7,5	7,5	0	0	-	7,5	7,5	7,5	7,5	0	0	-	7,5	7,5	7,5	-	172,5	3	0
	Worked	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	-	8	7,5	7,5	-	23		
	Overtime	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	-	0,5	0	0	-	0,5		
Šimon Ptáčník	Plan	W	W	W	W	W				W	W	W	W	W				W	W	W	W	W				W	HO	HO	HO	W				W	W	W				
	To work	-	-	-	-	-	-	-	37,5	-	-	-	-	-	-	-	37,5	-	-	-	-	-	-	-	37,5	-	-	-	-	-	-	-	37,5	-	-	-	22,5	172,5	0	0
	Worked	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
	Overtime	-	-	-	-	-	0	0	-	-	-	-	-	-	0	0	-	-	-	-	-	-	0	0	-	-	-	-	-	-	0	0	-	-	-	-	-	0		

Obrázek 8.7: Ukázka .xlsx souboru

Ve výpisu 8.16 najdeme, jakým způsobem je tvořena instance třídy `ExcelPackage`. Dále je zde ukázaná práce s buňkami budoucího .xlsx souboru. V tomto případě je buňce `A1` je nastavena textová hodnota „Employee“. Poté je nastavena barva fontu a barva pozadí pomocí tříd a jejich vlastností, které jsou součástí EPPlus knihovny.

```
ExcelPackage package = new ExcelPackage();
var reportSheet = package.Workbook.Worksheets.Add("Report");

reportSheet.Cells["A1"].Value = "Employee";
reportSheet.Cells["A1"].Style.Font.Color.SetColor(1, 255, 255, 255);
reportSheet.Cells["A1"].Style.Fill.BackgroundColor.SetColor(1, 0, 0, 104);
.
.
.
MemoryStream stream = new MemoryStream();
package.SaveAs(stream);
stream.Position = 0;
return File(stream, System.Net.Mime.MediaTypeNames.Application.Octet,"
    MonthlyReport"+Month.ToString("D2")+Year+".xlsx");
```

Výpis 8.16: Ukázka práce s EPPlus knihovnou

[2]https://www.epplussoftware.com/

Ve výpisu 8.17 se nachází LINQ dotaz, který zajišťuje vybrání plánů zaměstnance podle jeho identifikačního kódu `userId` a data `date`. Musely být vybrány pouze plány, které mají nastaven atribut `Activity.Type` v databázi na hodnotu „Absence“, ale zároveň bylo potřeba zajistit, aby se nebraly v potaz svátky, které mají v databází tento atribut nastaven na stejnou hodnotu. To je zajištěno pomocí `Priority`, kdy svátky jsou nastaveny na hodnotu 1. V poslední řadě jsme vybrali pouze plány, kde `ActivityType.Code` je nastaven na hodnotu „H“ nebo „HDH“.

```
var vacations = await db.UserWorkPlans.AsQueryable()
.Where(x => x.User.Id == userId)
.Where(x => (x.WorkPlan.StartDate.Value.Year == date.Year)
        && (x.WorkPlan.StartDate.Value.Month == date.Month))
.Where(x => x.WorkPlan.ActivityType.Type == ActionType.Absence)
.Where(x => x.WorkPlan.Priority != 1)
.Where(x => x.WorkPlan.ActivityType.Code == "H"
         || x.WorkPlan.ActivityType.Code == "HDH")
.Select(x => x.WorkPlan).ToListAsync();
```

Výpis 8.17: Ukázka dotazu pro vybrání dovolených z databáze

Výpis 8.18 ukazuje finální počítání dovolené. Do této metody jsou poslílány vybrané plány z předchozího výpisu 8.17. Plány se postupně v cyklu prochází, přičemž je kontrolována hodnota atributu `ActivityType.Code`. Pokud je nastavena na hodnotu „HDH“ přičte se k proměnné `VacationDays` 0.5, protože tato hodnota značí půldenní dovolenou. V opačném případě přičte 1.

```
public decimal CountVacationss(List<WorkPlan> vacations)
{
   decimal VacationDays = 0;
   foreach (WorkPlan vacation in vacations)
   {
    if (vacation.ActivityType.Code == "HDH")
    {
        VacationDays += 0.5;
    }
    else{ VacationDays++; }
   }
return VacationDays;
}
```

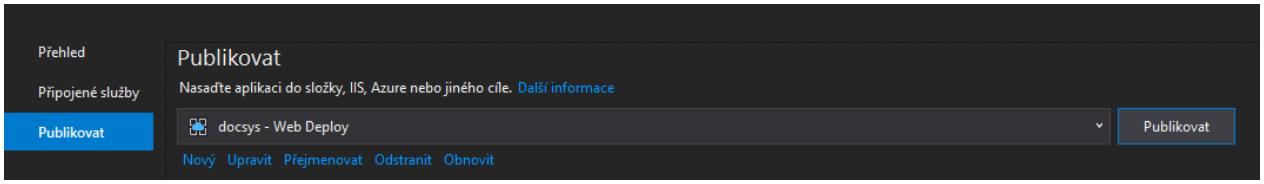
Výpis 8.18: Ukázka výpočtu dovolené

Kapitola 9

Zvěřejnění, testování a ladění

9.1 Zvěřejnění

Zveřejnění probíhalo pomocí nástroje **„Publikovat“**, který je součástí Microsoft Visual Studio (7.2). Tento nástroj umožňuje více možností, kam lze aplikaci zveřejnit. Jelikož jsem byl již přihlášen ke svému Azure účtu, kde jsem měl aktivovanou službu Web Sites (7.1.2). Zvolil jsem v tomto nástroji možnost `Azure`. Poté už mi nástroj automaticky nabídl zmiňovanou službu Web Sites. Dále bylo nutné potvrdit samotné zveřejnění (obrázek 9.1). Nakonec jsem poslal odkaz na webové stránky aplikace osobám, které se měly podílet na finálním testování.



Obrázek 9.1: Ukázka zveřejnění

9.2 Testování a ladění

Testování probíhalo jak mnou samotným, tak zaměstnanci firmy, od kterých jsem dostával pravidelnou zpětnou vazbu o nalezených nedostatcích. Tyto nedostatky jsem se poté snažil eliminovat změnou programového kódu aplikace.

Kapitola 10

Závěr

10.1 Znalosti uplatněné v průběhu praxe

Na začátku praxe mi hodně pomohly znalosti z předmětu Úvod do databázových systémů, které jsem použil při navrhování databázového modelu. V pozdější fázi jsem navrhoval obrazovky a jejich funkcionalitu, podobná náplň práce byla v předmětu s názvem Uživatelská rozhraní. Hlavní náplní mé práce bylo programování webové aplikace v jazyce .NET. Znalost tohoto jazyka mě z velké části naučily předměty Programovací jazyky II a Vývoj informačních systémů. Při tvorbě webové aplikace je důležité se orientovat v základech jazyka HTML, CSS a JavaScript. S těmito jazyky jsem se již setkal v předmětu Tvorba aplikací pro mobilní zařízení I.

10.2 Znalosti scházející v průběhu praxe

V průběhu praxe jsem se poprvé dostal do styku s platformou Microsoft Azure, kterou jsem využíval opakovaně po celou dobu. Z této platformy jsem obzvlášť využíval produkt s názvem Azure DevOps a ve finální fázi i Azure Web Sites. Druhou technologií, kterou jsem před nástupem neznal, byl rámec Entity Framework.

10.3 Celkové zhodnocení praxe

Praxe mi zanechala mnoho cenných zkušeností. Nejvíce oceňuji, že jsem si mohl vyzkoušet, jak funguje práce v týmu na reálném projektu od začátku až do konce. Dále mi praxe velmi pomohla ve zlepšení programátorských schopností a ukázala mi některé moderní technologie, které se dnes používají k vývoji.

Literatura

1. *ComProMiS* [online] [cit. 2021-04-18]. Dostupné z: https://www.compromis.cz/new.
2. *Zákoník práce (Česko, 2006)* [online] [cit. 2021-04-21]. Dostupné z: https://cs.wikipedia.org/wiki/Z%C3%A1kon%C3%ADk_pr%C3%A1ce_(%C4%8Cesko,_2006).
3. *EziData Solutions | Publish a VueJS or React Application to Azure* [online] [cit. 2021-04-18]. Dostupné z: http://blog.ezidata.com.au/post/Publish-a-VueJS-or-React-Application-to-Azure.aspx.
4. *Microsoft Visual Studio* [online] [cit. 2021-04-18]. Dostupné z: https://cs.wikipedia.org/wiki/Microsoft_Visual_Studio.
5. *Microsoft SQL Server* [online] [cit. 2021-04-18]. Dostupné z: https://en.wikipedia.org/wiki/Microsoft_SQL_Server.
6. *SQL Server Management Studio (SSMS) - SQL Server Management Studio (SSMS)* [online] [cit. 2021-04-18]. Dostupné z: https://docs.microsoft.com/cs-cz/sql/ssms/sql-server-management-studio-ssms?view=sql-server-ver15.
7. *ASP.NET MVC Framework* [online] [cit. 2021-04-18]. Dostupné z: https://cs.wikipedia.org/wiki/ASP.NET_MVC_Framework.
8. *C Sharp* [online] [cit. 2021-04-18]. Dostupné z: https://cs.wikipedia.org/wiki/C_Sharp.
9. *Razor Referenční informace k syntaxi pro ASP.NET Core* [online] [cit. 2021-04-18]. Dostupné z: https://docs.microsoft.com/cs-cz/aspnet/core/mvc/views/razor?view=aspnetcore-5.0&viewFallbackFrom=aspnetcore-3.1..
10. *Introduction to Identity - ASP.NET documentation* [online] [cit. 2021-04-18]. Dostupné z: https://aspnetcore.readthedocs.io/en/stable/security/authentication/identity.html.
11. *nager/Nager.Date* [online] [cit. 2021-04-19]. Dostupné z: https://github.com/nager/Nager.Date.
12. *Lekce 8 - Scaffolding a Entity Framework v ASP.NET Core MVC* [online] [cit. 2021-04-18]. Dostupné z: https://www.itnetwork.cz/csharp/asp-net-core/zaklady/scaffolding-a-entity-framework-v-aspnet-core-mvc.

13. *Code First initializers a migrace - kompletní přehled* [online] [cit. 2021-04-18]. Dostupné z: https://www.dotnetportal.cz/clanek/8475/Code-First-initializers-a-migrace-kompletni-prehled.

14. *Hypertext Markup Language* [online] [cit. 2021-04-18]. Dostupné z: https://cs.wikipedia.org/wiki/Hypertext_Markup_Language.

15. *Kaskádové styly* [online] [cit. 2021-04-18]. Dostupné z: https://cs.wikipedia.org/wiki/Kask%C3%A1dov%C3%A9_styly.

16. *Bootstrap* [online] [cit. 2021-04-18]. Dostupné z: https://cs.wikipedia.org/wiki/Bootstrap.

17. *Creately* [online] [cit. 2021-04-18]. Dostupné z: https://en.wikipedia.org/wiki/Creately.

18. *Odeslání e-mailu pomocí SendGrid s Azure* [online] [cit. 2021-04-22]. Dostupné z: https://docs.microsoft.com/cs-cz/azure/sendgrid-dotnet-how-to-send-email.